HLA OBJECT MODEL DEVELOPMENT: A PROCESS VIEW

Mr. Robert Lutz
Johns Hopkins University Applied Physics Laboratory
Laurel, MD

KEYWORDS
OMT, SOM, FOM, Process

ABSTRACT

One of the principle mechanisms for facilitating interoperability between simulations and reuse of simulation components within the High Level Architecture (HLA) paradigm is the object model. Federation Object Models (FOMs) are used to define the exchange of public data among the participants in an HLA federation. Simulation Object Models (SOMs) are used to describe the intrinsic capabilities that individual simulation systems can offer to HLA federations. The HLA Object Model Template (OMT) defines a common structure for describing the content and format of an HLA object model. Although the HLA OMT specification provides a complete description of each individual OMT component, there is little guidance in the OMT specification regarding the step-by-step process of HLA object model construction. The purpose of this paper is to suggest a generic, cookbook approach to the development of HLA object models. The major stages of HLA object model construction are discussed, along with a recommended sequence of developmental activities within each stage.

BACKGROUND

The Department of Defense (DoD) High Level Architecture (HLA) has been developed in response to the DoD Modeling and Simulation (M&S) Master Plan, which calls for a DoD-wide common technical framework which will apply to the full range of potential M&S applications. The objective of the HLA is to facilitate interoperability among simulations and promote reuse of simulations and their components.

The HLA object model is key to achieving these stated goals of interoperability and reuse, and represents one of the fundamental tenets upon which the HLA has been defined. Within the HLA paradigm, object models can take one of two forms. An HLA Federation Object Model (FOM) provides a specification of the exchange of public data among all of the participants in a HLA federation. An HLA Simulation Object Model (SOM), in contrast, provides a specification of the intrinsic capabilities that an individual simulation offers to federations. SOMs are utilized in the HLA Federation Execution Development Process (FEDEP) as a means of determining the suitability of individual simulations to participate in an HLA federation.

The standard presentation format and content for both FOMs and SOMs is provided by the HLA Object Model Template (OMT). The categories of information described by the OMT collectively defines the information model which is required of all HLA federations and federation participants. The HLA OMT Extensions provide supplementary categories of optional information which may provide additional clarity and completeness in some types of object models. A short overview description of each OMT and OMT Extensions component is provided below. More detailed descriptions and graphical illustrations of table formats are provided in the OMT and OMT Extensions documents.

OMT COMPONENTS

***Object Class Structure Table*:** provides the template for recording the namespace of all public object classes within a given simulation or federation domain. The structure of the template also supports the specification of hierarchical relationships between low-level, instantiable object classes and their more abstract, higher-level superclasses. The specification of a particular HLA object class hierarchy is driven by attribute inheritance and the subscription requirements of federation participants.

***Object Interaction Table*:** provides the template for recordation of interactions between public object classes. An interaction is an explicit action taken by an object, that may potentially affect the state of another object within the federation. Interactions are specified in HLA object models in terms of an interaction class hierarchy, classes of initiating and receiving objects (and their affected attributes), and required parameters.

***Attribute/Parameter Table*:** provides the template for describing and recording the set of attributes which characterize each public object class. In addition, this table is used to characterize the parameters of interaction classes. Associated with this table are two additional subtables used to characterize user-defined (enumerated and complex) datatypes.

***FOM/SOM Lexicon*:** provides a structured format for defining all terms (classes, interactions, attributes, parameters, ...) used in an HLA object model.

OMT EXTENSIONS COMPONENTS

***Component Structure Table*:** provides the template for recordation of "part-whole" relationships within a simulation or federation domain. A part-whole relation between two classes indicates that objects from one class are parts (or components) of composite objects from another class. An HLA component structure is simply a set of these part-whole (or component) relations along with their cardinalities.

***Associations Table*:** provides the template for recordation of associations between object classes other than part-whole relationships. This component of a HLA object model is designed to capture those static object relationships which are considered necessary for assessments of interoperability and reuse.

***Object Model Metadata*:** provides a means to specify information about a federation or simulation as a whole. Such information is considered critical to determining the reuse potential of a particular FOM, and is also useful for describing the full range of applicable characteristics of a potential federate. The format

for describing this type of information is simple text (no explicit formatting constraints).

OM DEVELOPMENT PROCESS

Although the OMT and OMT Extensions together fully define the format and content of an HLA object model, neither of these documents provide any significant user guidance on the process of object model construction. One possible analogy is that of a hardware systems developer being given the detailed specifications to all of the low-level components of a complex system, but not being given any information as to how the individual components can be best assembled into a working whole. The systems developer in situations like this will many times be able to "figure it out", based on experience and trial-and-error experimentation. However, the availability of early guidance describing a logical sequence of activities for constructing the system (from individual subsystem development to full system development) is likely to result in a much more timely and cost-efficient development process than would be otherwise possible. Likewise, for HLA object model developers, although all of the components required for object model construction are fully defined in the OMT specification, the availability of a common process view (which delineates the logical sequence of activities necessary for effectively utilizing the OMT components) would strongly facilitate efficient object model development practices throughout the HLA community.

The purpose of the following sections are to describe a generalized process for construction of both Federation and Simulation Object Models. This process is expected to be applicable across most HLA applications. Before proceeding, it should be noted that this suggested sequence of development activities is not the only process which can lead to efficient and robust object model construction. In fact, many deviations from this process are possible which can lead to successful results.

It is also important to note that there are several classes of information which are relevant to achieving interoperability within an HLA federation that are not explicitly supported in the OMT. Examples of these categories of information includes agreements on common databases/algorithms, specification of object interaction sequences, and documentation of security procedures. While recognized as meaningful to facilitating interoperability and reuse, such considerations are considered to be supplemental (external) to the information provided in an HLA object model. Nonetheless, the process view discussed in this paper will highlight these types of related issues whenever they are considered to be pertinent to the object model development process.

Throughout the discussions in this paper, the reader will note that there is great flexibility given to object model developers regarding the names given to data elements in an object model. While this accurately reflects the current guidance, it is expected that evolving DoD-wide standards on class/attribute/parameter naming conventions and associated semantics will lead to significant improvements and efficiencies in the object modeling process. Thus, although the process descriptions in this paper suggest significant latitude in naming conventions, the reader should understand that the longer-term trend is toward standardization.

Simulation Object Model Development

In the HLA Rules, the first rule for individual federates (Rule 6) states that all federates shall have an HLA SOM, documented in accordance with the HLA OMT. Thus, all simulations deemed to have "lasting value" to a DoD agency or organization are required to be described by a SOM as a prerequisite to achieving HLA compliance. Besides facilitating reuse of the simulation in future federations, SOMs also represent an important building block in the HLA FOM development process. Because of this inherent dependency, this paper will focus first on the process of SOM development.

Although it is considered possible to describe the process of HLA FOM development in terms of a generalized approach with broad community applicability, a process description for HLA SOM development is less likely to be general in nature. The primary reason is that the various agencies of the DoD M&S community are composed of many different "cultures" of software engineering practices. One aspect of these "cultures" are established methodologies for traditional object model development. While many of these communities have shown themselves to be willing to map their internal methodologies and practices to a common process view for the purpose of interoperating with other simulation systems, these same communities may be less likely to abandon established methodologies for the purpose of

simply describing their intrinsic functionalities. Still, some general guidelines and useful advise is possible, which can prove beneficial to SOM developers for whatever specific development process is chosen.

The starting point the following process description is a simulation concept. Note that having working software or even an existing software design is not a prerequisite for SOM construction. In fact, completing a SOM prior to detailed simulation design is considered advantageous since the SOM is, in a way, a requirements specification that describes how the simulation will interoperate with other simulation systems in the future. These interoperability and reuse requirements can be addressed from within the software design process, and the required functionalities built directly into the simulation rather than having to retrofit these capabilities subsequent to implementation (a more complex, costly, and time consuming process). Since however a SOM is intended to provide a specification of the current capabilities of a given simulation system, SOMs should not be made public until the functionality specified in the SOM has been fully instantiated.

The following process description provides a suggested sequence of activities for HLA SOM development. This process should be applicable for both new or existing simulations. An illustration of this process is provided in Figure 1.

Step 1: *Determine Publishing Capabilities for Object/Interaction Classes*

In this step, the classes that the simulation system can take responsibility for publishing during an HLA federation execution are identified. The determination of these classes is driven by the specific intrinsic functionalities that the simulation sponsor/developer considers to be useful to and can make available to future (presently unidentified) federations. The object classes which are identified as publishable by the simulation should generally correspond to some real world entity (or collection of real world entities). Each object class should be published at the level of generalization for which it can be instantiated within the simulation context. For instance, some simulations may provide explicit software representations of very specific systems, in which case the simulation may publish object classes like M1_TANK, F-16, or CG-47. Other simulations may have more data driven representations of systems, and support more generalized object classes for publication such as TANK, AIRCRAFT, or SHIP. Still others may support even more generalized, instantiatable classes such as GROUND_VEHICLES or AIR_VEHICLES. Object classes which are elements of an internal object class hierarchy that are purely abstract in nature (and thus cannot be instantiated) should not be considered publishable.

Public interaction classes are those explicit actions which are initiated by one or more public object class(s), that may have an affect or impact on

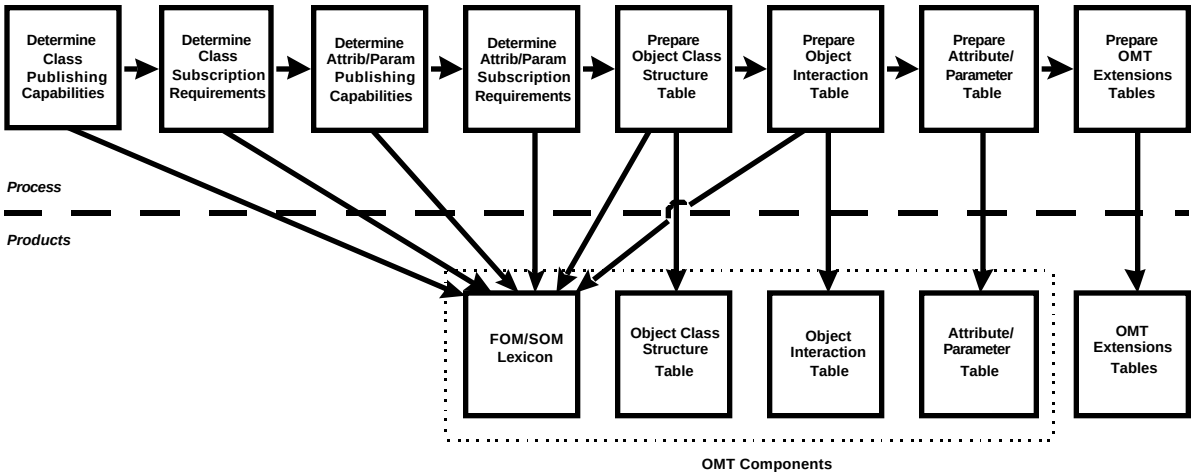

Figure 1 - SOM Development Process

objects in different federates. The SOM developer must consider the potential behaviors of its publishable objects, and determine the events these objects may generate that may be of potential interest to other federates. Once all public interaction classes have been determined, the object classes which can initiate each type of interaction should also be identified for later inclusion in the Object Class Structure Table.

Neither object classes or interaction classes necessarily need to map to an identifiable class within the simulation itself. In fact, this would not be possible for non-object-oriented simulations. However, it is the responsibility of each simulation to provide a software mapping between its internal functionalities and its external public view.

SOM developers are free to name their classes anything they prefer. The namespace of all publishable classes should be documented in the FOM/SOM Lexicon at the conclusion of this step.

Step 2: *Determine Subscription Requirements for Object/Interaction Classes*

In addition to specifying the classes of data a simulation can export, the SOM is also intended as a means of specifying the types of data that a simulation may want to import from other federates. There are two basic categories of imported class data. First, there are those object and/or interaction classes that a federate can reflect in their simulation as well as publish themselves. For instance, a simulation may be capable of publishing aircraft objects, but may also have the capability of reflecting instances of aircraft objects owned by other federates. The other category of imported class data are those object and/or interaction classes that a federate cannot publish by themselves, but can be reflected in the federate's simulated environment if other federates produce the data. For instance, a given naval simulation may not explicitly represent ground forces, but may be able to shoot at ground forces reflected from other federates if that data was made available. In addition, all object classes which will act as recipients of imported interactions must be identified in this step.

As with publishable classes, the names of all imported (subscribable) classes are at the discretion of the SOM developer. These classes should be identified and documented in the FOM/SOM Lexicon.

Step 3: *Determine Publishing Capabilities for Attributes/Parameters*

In this step, all classes identified as "publishable" are characterized according to an identified set of attributes or parameters. For object classes, this entails specifying the complete set of public object characteristics that can be explicitly supported by the simulation, and are considered to be potentially useful in future federations. The number (and types) of attributes that are supported for a given class is generally related to the degree of fidelity provided by the underlying model.

For interaction classes, this step includes specifying the complete set of parametric information that can be provided by the simulation for interactions it can initiate. It is the responsibility of the SOM developer to anticipate what parameters recipients of the interaction will need to calculate the associated effects, and also to make certain that, besides specifying this data in the SOM, the simulation can reasonably provide the required data.

At the conclusion of this step, all attributes and parameters identified as publishable should be documented in the FOM/SOM Lexicon. Attribute/parameter names are selected by the SOM developer.

Step 4: *Determine Subscription Requirements for Attributes/Parameters*

In this step, all classes identified as "subscribable" are characterized according to an identified set of attributes or parameters. For object classes, the attributes identified are those which have "semantic meaning" within the context of the simulation. For instance, if LOCATION is the only useful characteristic of a ground force representation reflected by a naval simulation (perhaps because the only meaning of a ground force entity in the naval simulation is as a target location), then LOCATION would be the only attribute of this class. If this ground force could also be detected by a radar system, then ORIENTATION and SIGNATURE may perhaps also represent useful attributes. Only that attribute-level information which the simulation can make substantive use of should be specified.

For interaction classes, this step should entail the specification of all information that needs to be provided with externally-initiated interactions. For

instance, subscribers of a WEAPON_DETONATE interaction may need to know information about the location and/or orientation of the weapon at detonation, the type and size of the warhead, and perhaps additional information as well depending on how the simulation processes the occurrence of the interaction. As with attributes, only those interaction parameters which the simulation can make substantive use of should be specified.

At the conclusion of this step, all attributes and parameters identified as uniquely subscribable should be included in the FOM/SOM Lexicon. Attribute/parameter names are selected by the SOM developer.

Step 5: *Prepare Object Class Structure Table*

In this step, the namespace of all object classes identified in the lexicon are mapped to a class hierarchy in the Object Class Structure Table. Although the subscription requirements of individual federates tends to drive the structure of class hierarchies at the federation level, the subscription requirements of future federations will be unknown at the time of SOM construction. For SOMs, there are (at least) three viable reasons for specifying classes other than leaf nodes in a class hierarchy. The first reason is that some higher level classes may themselves be publishable or subscribable. For instance, a simulation may have the ability to publish or subscribe to a generalized PLATFORM class, which can be cast as a superclass to more specific TANK, AIRCRAFT, and SHIP classes. A second reason may be to utilize attribute inheritance as a shorthand means of specifying common attributes of several publishable classes, in which case the resulting abstract class would be designated as neither publishable or subscribable. The third primary reason may be to show some or all of an internal class hierarchy, even if only the concrete classes are publishable or subscribable. This last reason, while permitted, tends to violate the HLA tenet that a SOM provides only the external public view of the simulation's capabilities, and is therefore discouraged.

For simulations that have no publishable or subscribable classes beyond that at the leaf level (no identifiable publishable or subscribable superclasses), a completely flat class structure is perfectly valid, and may even have advantages when reconciling object class structures at the federation level. It is the decision of each SOM developer as to whether the specification of a multi-level class hierarchy makes sense for their application.

Step 6: *Prepare Object Interaction Table*

In this step, the information content defined by the Object Interaction Table is fully documented. As with object classes, the specification of a multi-level hierarchy of interaction classes will only be useful in certain situations, with flat structures being perfectly appropriate in many cases. The identity of object classes which can serve as recipients of published interactions or as initiators of subscribed interactions will be unknown for SOMs, since they are external to the simulation's local environment. In this case, temporary names based on the known context of the interaction class can and should be provided.

Step 7: *Prepare Attribute/Parameter Table*

In this step, the attributes and parameters identified in Steps 3 and 4 are documented in the Attribute/Parameter Table. The description of these attributes/parameters in the table should be based on what the simulation can currently support (for publishable data), or on what attribute/parameter characteristics are required (for subscribable data). All user-defined datatypes should be fully characterized in the Enumerated or Complex Datatype Table.

Step 8: *Prepare OMT Extensions Tables*
(Optional)

In this step, the components of the OMT Extensions are considered for inclusion in the object model. The Object Model Metadata is of special importance for SOMs, since it provides a means (beyond the information captured in the OMT tables) to "advertise" the unique features and capabilities the simulation can offer to future federations. For instance, SOM developers may want use the Object Model Metadata as a means to highlight key algorithms, VV&A histories, security characteristics, and any other information which may be of interest to future federation developers, but which cannot be directly captured in the OMT tables.

The inclusion of the Associations and Component Structure Tables should be considered whenever explicitly specifying these types of object relationships conveys a deeper understanding of how the simulation is designed. For instance, the

Component Structure Table is quite useful in engineering simulations for describing assemblies of complex systems, and the Associations Table is useful for representing chains of command in applications where explicit C^3 software representations are provided. Object class references in these tables should always correspond to an entry in the Object Class Structure Table.

Federation Object Model Development

The starting point for the HLA FOM development process is the Federation Development phase of the HLA FEDEP model. Implicit to this assumption is that the federation requirements have been clearly delineated, the scenario(s) have been defined, the real world objects and interactions that are to be represented explicitly in the federation have been determined, and the federation participants have been both identified and fully described according to an HLA SOM. The sequence of activities for HLA FOM development are illustrated in Figure 2, and are described in the following sections.

Step 1: *Determine Federate Publishing Capabilities and Subscription Needs*

In this step, each federation participant maps their SOM to the conceptual objects and interactions in the federation to determine which classes each individual federate is capable of publishing, and which classes each federate will need to subscribe to. Each federate will need to map the namespace of their SOM classes to the naming conventions used at the conceptual level, and determine if there are any semantic differences which preclude assuming responsibility for publishing certain classes.

The output of this step is a cross-matrix for each federate which maps individual object and interaction classes from each federate's SOM to the required conceptual objects and interactions at the federation level. Each entry in the matrix should either be a "P" for the ability to publish, a "S" for the need to subscribe, or "PS" for both. Alternative formats for this data are acceptable, but should be common across any given federation.

Step 2: *Determine Federation Publishing Responsibilities*

In this step, the namespace of the federation's object and interaction classes is determined, along with identifying which federates will assume responsibility for each class. One popular means by which this may be accomplished is to arrange a structured object model development meeting (affectionately termed a "FOMorama") led by a volunteer coordinator. In this meeting, each conceptual object/interaction is discussed one at a time to determine which of the federates are willing to assume publishing responsibilities for each class, and which federates will need to subscribe to each class. All object classes which can act as initiators or recipients of interactions should be identified as such. Classes for which there are no subscribing federates can be removed from further consideration. Agreements on the name and semantics for each class must be negotiated among the federates. Other relevant issues for discussion include level of fidelity, individual object behavior, and environmental representation.

The output of these discussions is documented via the FOM/SOM Lexicon, which identifies the name of each object and interaction class that is to be supported in the federation, along with its specific meaning. A table or matrix which captures the

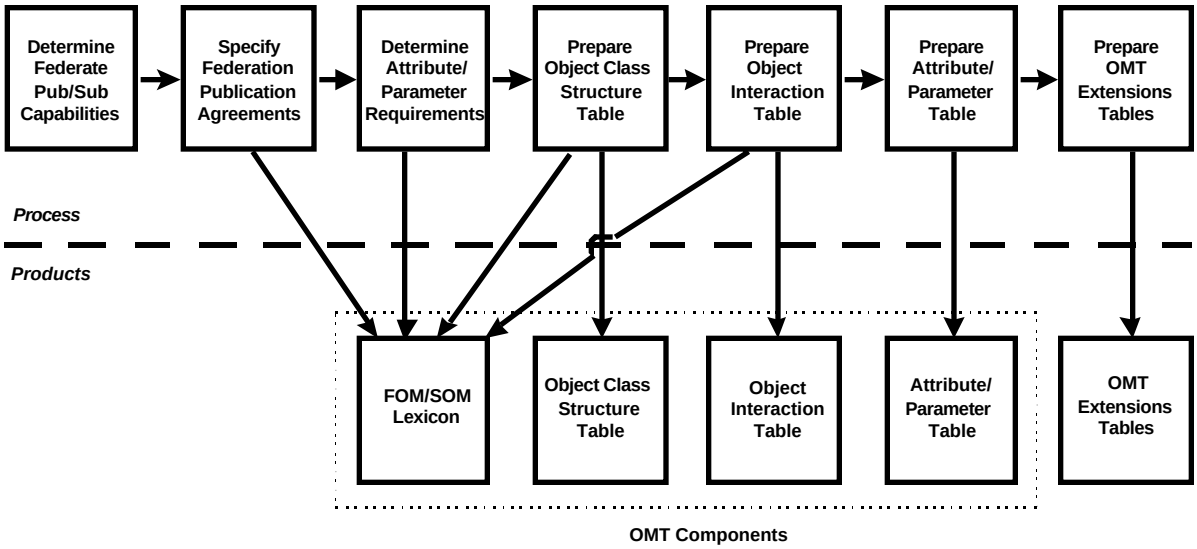

Figure 2 - FOM Development Process

federation-wide agreements on publishing responsibilities and subscription needs should also be prepared.

Step 3: *Determine Attribute and Parameter Requirements*

In this step, the attributes of all public object classes and the parameters of all object interactions are identified. To realize this, the meeting coordinator first revisits each individual object class, and polls the subscribers of the class as to the specific attribute-level information that they require. This set of information is then compared to the set of attributes that the class publishers can currently support. This results in one of three possibilities per attribute:

- For attributes that have publishers but no subscribers, that attribute is removed from further consideration in the FOM since it does not need to be exchanged publicly.
- For attributes that have subscribers but no publishers, either software modifications will need to be defined which permits the publishing federate(s) to explicitly support the attribute (in which case the attribute would appear in the FOM) or the subscribing federate(s) would need to make whatever software modifications or other actions that are necessary to be able to participate in the federation without this information (in which case the attribute would not appear in the FOM).
- For attributes in which publisher/subscriber matches are found, the semantics and fidelity requirements of the attribute should be discussed to determine if a "match" really exists. If so, the name of the attribute (for the purposes of the federation) should be defined.

A similar process should be followed for interaction parameters. First, subscribers of the interaction should dictate their requirements for interaction parameters, based on the information required by the federate to calculate the effects of the interaction on the state of objects it "owns". This is compared to the information, in the form of parameters, that interaction class publishers can provide. The output of this mapping (per parameter) will result in one of the same three possibilities shown above for attributes, and should be handled in the same fashion.

The output of this step should be the identification of all attributes and parameters that are to be supported across the federation, along with their associated semantics. This information is documented in the FOM/SOM Lexicon. Required software modifications to participating federates should also be noted for future reference.

Step 4: *Prepare Object Class Structure Table*

In this step, the namespace of public object classes defined in the FOM/SOM Lexicon are mapped to a class hierarchy which meets the needs of the federation. As is stated in the HLA OMT, the two main considerations in defining a FOM class structure are attribute inheritance and subscription requirements. One possible procedure for building this class structure is to begin by forming clusters of the real world (concrete, or instantiatable) classes that appear to have some semantic linkages. For instance, tanks, aircraft, and ships all represent "things that move", while guns and TELs represent "things that shoot". These clusters all offer opportunities to define abstract classes that represent the elements of the cluster at a more general level. Whether or not an abstract class should be defined for the cluster depends on 1) whether any attribute-level information can and should be "pushed up" to the abstract class, and 2) whether any federate would find it useful to subscribe to the abstract class rather than each individual member of the cluster. In this latter case, this would depend on whether 1) the federate needs to know the identity (concrete class name) of each instance of any or all classes in the cluster and 2) whether sufficient attribute-level information can be defined at the level of the abstract class without causing the inheritance of unnecessary information at the level of the concrete class.

If an abstract class appears to be beneficial for a cluster, that class should be named and fully defined in the FOM/SOM Lexicon. Also, the attributes that are defined for the abstract class should be noted, and removed (if applicable) from all lower-level classes which will inherit this information.

In a similar fashion, the abstract classes that have been defined at this point may themselves may be grouped into semantically similar clusters, to determine if higher levels of abstraction are appropriate. This "bottom-up" approach can be used recursively to generate the structure of the entire class hierarchy, which may have one or more roots. Other object modeling approaches (including

top-down) may also have merit in some situations; the actual methodology for defining the class-subclass relationships among public object classes is entirely at the discretion of the federation developers.

As the final activity in this step, the class hierarchy is fully documented in the Object Class Structure Table, and the publish/subscribe designations are associated with each class in the table. As is noted in the OMT, concrete classes are always designated as "publishable" and "subscribable", while abstract classes are designated as either "subscribable" or as "neither".

Step 5: *Prepare Object Interaction Table*

In this step, the information categories defined in the Object Interaction Table are fully defined. The first activity in this step is to map the namespace of interaction classes defined in the FOM/SOM Lexicon to a hierarchical structure. Here, since the OMT specification states that inheritance of interaction parameters is not permitted, only federation subscription requirements need to be considered in defining the structure. A bottom-up approach as discussed above is suggested, but certainly not required. Also, it is important to note that in some situations, entirely flat structures (no abstract classes) are considered perfectly valid. The need to define abstract classes (for either objects or interactions) is totally dependent on the preferences and needs of the federation participants.

The next activity is to specify the interaction parameters associated with each concrete interaction class. This can be copied directly from the FOM/SOM Lexicon as a result of activities to define these parameters in Step 3.

The next activity is to define the initiating and receiving classes for each concrete interaction class, along with the specification of all attributes that may be potentially affected by the interaction. Publishing federates are responsible for completing the "Initiating Object" column, while subscribing federates should indicate the receiving objects and affected attributes. All class and attribute names should have existing references in the FOM/SOM Lexicon.

The next activity is to complete the "Init/Sense/React" column in the Object Interaction Table. As is stated in the OMT, the designation for each interaction should be either "Init/React" or "Init/Sense" depending on whether at least one federation participant can actively react to the interaction (eg., by modifying the value of one or more affected attribute), or whether all subscribers of the interaction react entirely passively (eg., by simply logging the information).

Finally, if all of the information defined in this step has not yet been captured directly in the Object Interaction Table, this table must now be fully completed. It is also at this time that agreements on object interaction sequences and trigger conditions must be agreed to across the federation, and documented in a suitable format. Other means of specifying federation behavior as defined by the federation scenario may also be applicable.

Step 6: *Prepare Attribute/Parameter Table*

In this step, all of the object attributes and interaction parameters defined in the FOM/SOM Lexicon must be documented in the Attribute/Parameter Table, along with the associated characteristics defined for the table. This requires addressing each attribute and parameter individually, negotiating between publishers and subscribers relevant issues such as accuracy, resolution, update type, and update rate (if applicable). The results of these negotiations may result in required software modifications, which must be noted for later reference. Decisions on bundling sets of attributes which must be updated as a "package" is also done at this time.

For user-defined attribute/parameter datatypes, one or both of the special subtables may be needed. For complex datatypes, the same publisher/subscriber negotiations required for singular attributes will be required for each individual field. For enumerated datatypes, the enumerations themselves (and associated integer representations) must be agreed to across the federation.

For each attribute, the table requires that an indication is provided as to whether the ownership of the attribute may be transferred to another federate during execution. If the federation requires this for any given attribute, a description of the ownership transfer conditions and rules should be documented. The OMT "notes" feature is useful for this purpose.

Finally, security considerations may require that each attribute/parameter be designated at an appropriate level of classification. Although this is not currently supported by the OMT specification,

the Attribute/Parameter Table may be easily augmented with the appropriate information.

Step 7: *Prepare OMT Extensions Tables (Optional)*

In this step, the components of the OMT Extensions are considered for inclusion in the object model. At a minimum, this should include providing appropriate descriptive data about the federation via the fields of the Object Model Metadata. In addition, as with SOMs, the Associations Table and Component Structure Table should be considered for inclusion whenever the associated data enhances the clarity and understanding of the object model within the given application. The decision as to the appropriateness of the OMT Extensions Tables for a particular application is at the discretion of the FOM developers.

SUMMARY

The process descriptions provided in this paper have been intended to provide HLA object model developers with a baseline operational framework for FOM or SOM development. These process descriptions are expected to evolve and mature over time, as developers continue to accrue experience in this particular phase of HLA federation development. Although not explicitly highlighted in this paper, the DMSO-sponsored Object Model Development Tools (OMDTs), currently in alpha testing, will be tightly integrated into the FOM/SOM development process in the future.

REFERENCES

All references are available for download via the home page of the Defense Modeling and Simulation Office, site address http://www.dmso.mil.

[1] Under Secretary of Defense for Acquisition and Technology, "Department of Defense Modeling and Simulation Master Plan, DoD 5000.59-P," October 1995.

[2] Defense Modeling and Simulation Office, "HLA Federation Development and Execution Process (FEDEP) Model, Version 1.0," 21 August 1996.

[3] Defense Modeling and Simulation Office, "HLA Object Model Template, Version 1.0," 21 August 1996.

[4] Defense Modeling and Simulation Office, "HLA OMT Extensions, Version 1.0," 21 August 1996.

[5] Defense Modeling and Simulation Office, "HLA Rules, Version 1.0," 15 August 1996.

[6] Lutz R., Hooks, M., Hunt K., "Automation in the HLA FOM Development Process," 15th DIS Workshop, 16-20 September 1996.